Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/KR05/000668

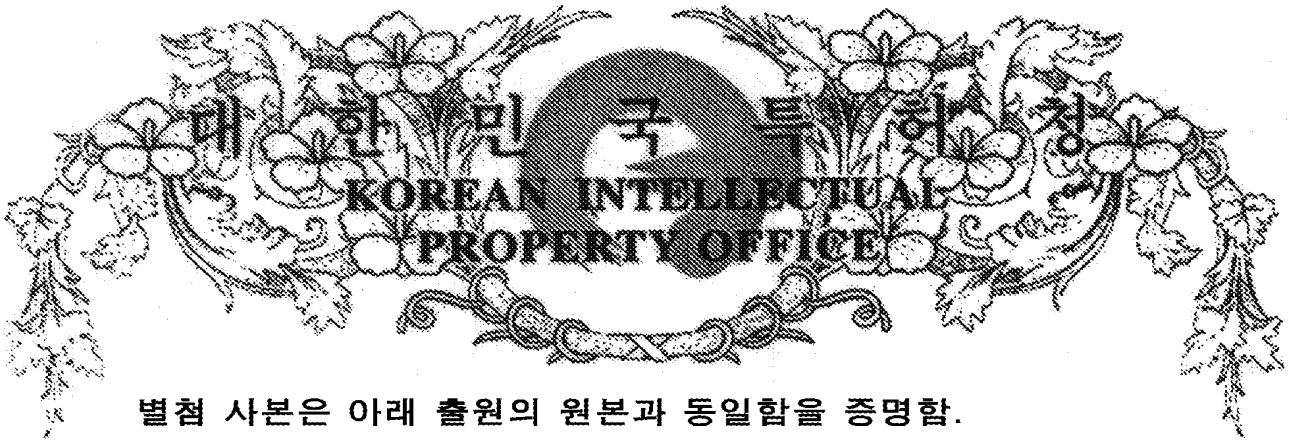
International filing date: 09 March 2005 (09.03.2005)

Document type: Certified copy of priority document

Document details: Country/Office: KR
Number: 10-2004-0055329
Filing date: 12 July 2004 (12.07.2004)

Date of receipt at the International Bureau: 30 June 2005 (30.06.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)



별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office

출 원 번 호 : 특허출원 2004년 제 0055329 호
Application Number 10-2004-0055329

출 원 일 자 : 2004년 07월 12일
Date of Application JUL 12, 2004

출 원 인 : 양세양
Applicant(s) YANG, Sei Yang

2005 년 06 월 09 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】 특허출원서

【권리구분】 특허

【수신처】 특허청장

【참조번호】 0001

【제출일자】 2004.07.12

【발명의 국문명칭】 재활용 기법을 채용한 고성능 설계검증 장치를 이를 활용한 신속한 설계검증 방법

【발명의 영문명칭】 High Performance Design Verification Apparatus Using Re-use Technique and Its Rapid Verification Method Using the Same

【출원인】

【성명】 양세양

【출원인코드】 4-1998-037998-4

【발명자】

【성명】 양세양

【출원인코드】 4-1998-037998-4

【취지】 특허법 제42조의 규정에 의하여 위와 같이 출원합니다. 출원인 양세양 (인)

【수수료】

【기본출원료】	0 면	38,000 원
【가산출원료】	23 면	23,000 원
【우선권주장료】	0 건	0 원
【심사청구료】	0 항	0 원
【합계】	61,000 원	
【감면사유】	개인(70%감면)	

【감면후 수수료】 18,300 원

【첨부서류】 1.요약서 · 명세서(도면)_2통

【요약서】

【요약】

본 발명은 설계된 수백만 게이트급 이상의 디지탈 회로의 설계 검증을 위한 설계 검증 장치 및 이를 이용한 설계 검증 방법에 관한 것이다.

본 발명에서는 임의의 설계검증 툴 내지는 설계검증 시스템에서 수행되어지는 본 발명의 설계검증 시스템소프트웨어로 하여금 설계검증 대상 회로에 존재하는 설계객체들 간에 동적정보 교환을 가능하게 하여 설계가 변경된 후에도 최대한 이전에 설계검증 실행의 결과들을 재활용하여서 신속한 설계 검증을 가능하게 한다.

【대표도】

도 1

【명세서】

【발명의 명칭】

재활용 기법을 채용한 고성능 설계검증 장치를 이를 활용한 신속한 설계검증 방법{High Performance Design Verification Apparatus Using Re-use Technique and Its Rapid Verification Method Using the Same}

【도면의 간단한 설명】

<1> 도1 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 시뮬레이터를 갖는 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 일 예를 개략적으로 도시한 도면.

<2> 도2 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 시뮬레이터를 갖는 2 이상의 컴퓨터들과 이들 컴퓨터들을 연결하는 컴퓨터 네트워크로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면.

<3> 도3 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 시뮬레이터를 갖는 컴퓨터와 하드웨어검증플랫폼으로 구성된 본 발명에 관한 설계 검증 장치의 일 예를 개략적으로 도시한 도면.

<4> 도4 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 시뮬레이터를 갖는 2 이상의 컴퓨터들과 하드웨어검증플랫폼과 이들 컴퓨터들을 연결하는 컴퓨터 네트워크로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면.

<5> 도5 는 설계객체가 수정된 후에도 설계객체가 수정되기 전에 실행된 설계검증 과정에서 얻어진 동적정보를 재활용하여서 설계객체가 수정된 후에 실행되는 설계검증 과정을 신속하게 진행하는 과정을 개략적으로 도시한 도면.

<6> 도6 은 설계객체가 수정된 전의 설계검증 실행 결과와 설계객체가 수정된 후의 설계검증 실행 결과가 최초로 달라지는 시점을 설계객체가 수정되기 전에 실행된 설계검증 과정에서 얻어진 1 이상의 동적정보를 병렬적으로 이용하여 신속하게 구하는 과정을 개략적으로 도시한 도면.

<7> <도면의 주요부분에 대한 부호의 설명>

<8> 27 : 하드웨어검증 플랫폼

<9> 32 : 검증 소프트웨어 34 : 시뮬레이터

<10> 35 : 컴퓨터

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<11> 본 발명은 설계된 수백만 게이트급 이상의 디지탈 시스템의 설계를 검증하는 기술에 관한 것으로, 설계된 수백만 게이트급 이상의 디지탈 시스템을 시뮬레이션, 시뮬레이션 가속, 하드웨어 에뮬레이션 등을 통하여 검증하고자 하는 경우에 검증의 성능와 효율성을 증가시키는 검증 장치 및 이를 이용한 검증 방법에 관한 것이다.

<12> 최근에 집적회로의 설계 및 반도체 공정기술이 급격하게 발달함에 따라 디지탈 회로 설계의 규모가 최소 수백만 게이트급에서 수천만 게이트급까지 커짐은 물론 그 구성이 극히 복잡해지고 있는 추세이고, 이와 같은 추세는 계속적으로 확대되고 있는 추세로 가까운 미래에 일억 게이트급 이상의 설계도 예상되고 있다. 그러나, 시장에서의 경쟁은 더욱 더 치열해지므로 빠른 시간 내에 우수한 제품을 개발하여야만 함으로 빠른 시간 내에 자동화된 방법으로 설계된 회로를 효율적으로 설계 검증하기 위한 효과적인 방법의 필요성이 더욱 커지고 있다.

<13> 지금까지는 설계된 디지탈 회로를 설계 검증하기 위하여서 하드웨어 기술언어(Hardware Description Language, 앞으로 이를 HDL로 약칭함)나 시스템기술언어(System Description Language, 앞으로 이를 SDL로 약칭함)를 이용한 설계 초기에는 소프트웨어적 접근법인 HDL/SDL 시뮬레이터들(예로 Verilog 시뮬레이터, VHDL 시뮬레이터, SystemC 시뮬레이터, 또는 SystemVerilog 시뮬레이터 등)이 주로 사용되어지고 있다. 이와 같은 시뮬레이터는 설계 검증 대상회로를 소프트웨어적으로 모델링한 순차적인 인스트럭션 시퀀스로 구성된 소프트웨어 코드를 컴퓨터 상에서 순차적으로 수행하여야 함으로 상기 수백만 게이트급 이상의 설계에 대해서는 시뮬레이션 성능의 저하가 설계대상의 크기에 비례하여 발생하는 것이 문제가 되고 있다. 일 예로 시스템온칩(System On a Chip, 앞으로는 이를 SOC로 약 칭함) 설계에서와 같이 1000만 게이트급 이상의 설계를 HDL/SDL 시뮬레이터로 시뮬레이션하는 경우에 현존하는 제일 빠른 프로세서를 장착한 컴퓨터에서 해당 HDL/SDL 시뮬레이터로 설계를 시뮬레이션하는 경우에 시뮬레이션 속도는 레지스터전송수준(Register

Transfer Level, 앞으로 이를 RTL로 약칭함)으로 하는 경우에 10 cycles/sec를 넘기기 어려우며, 게이트 수준에서 시뮬레이션을 진행하면 1-2 cycles/sec를 넘기기가 어려운 것이 매우 일반적이다. 그러나 해당 설계를 검증하고자 필요한 시뮬레이션 사이클은 최소 수백만 사이클에서부터 최대 수십억 사이클이 필요함으로 전체 시뮬레이션 시간은 상상을 초월하게 된다. 뿐만 아니라, 최근에는 이와 같은 복잡한 디지털 시스템 설계를 시뮬레이션을 이용하여 검증하는 경우에 모든 설계 오류들을 발견하여 제거할 수 있는 테스트벤치를 작성하는 것에 대한 어려움이 증대하고 있다.

<14> 이와 같은 장시간의 검증 시간의 단축과 테스트벤치 작성의 어려움을 해소시키기 위하여 현재 사용되는 기법들은 다음과 같은 것들이 있는데, 첫째는 하드웨어 기반의 검증 시스템(예로 시뮬레이션가속기, 하드웨어 에뮬레이터, 프로토타이핑시스템 등)을 사용하거나, 둘째는 특성검사(property checking) 혹은 모델검사(model checking) 내지는 등가검사(equivalence checking)과 같은 정식검증(formal verification)을 사용하는 것이다. 그러나 하드웨어 기반의 검증 시스템을 사용하는 것은 설계 초기에는 적용이 불가능하고, 합성이나 컴파일 과정이 HDL/SDL 시뮬레이터를 사용하는 것보다 오래 걸리며, 사용하기가 HDL/SDL 시뮬레이터에 비하여 매우 어렵고, 시스템의 구입 비용과 유지/보수 비용이 매우 클 뿐만 아니라, 무엇보다도 설계자나 검증엔지니어들이 HDL/SDL 시뮬레이터에 대한 선호도(특히 HDL 시뮬레이터에 대한 선호도)가 이들 하드웨어 기반의 검증 시스템에 비하여 매우 높고, HDL/SDL 시뮬레이터로는 아무 문제 없이 수행이 되는 설계 코드들이 하드웨

어 기반의 검증 시스템에서는 수행되지 않는 경우가 많아서 이들 하드웨어 기반의 검증 시스템들은 제한적인 상황과 제한적인 사용자들에서만 사용되고 있다. 뿐만 아니라, 이들 하드웨어기반의 검증시스템들은 시뮬레이터와 같은 수준의 제어도(controllability)와 관측도(observability)를 제공하지 못하고 설계검증대상(DUV: Design Under Verification, 이 후로는 DUV로 약칭함)에 존재하는 모든 신호선들을 탐침하게 되면 수행속도가 2배에서 10배 이상 늘어나는 문제점도 가지고 있다. 또한 정식검증 툴을 이용하는 것은 등가검사이외에 특성검사나 모델검사를 적용하기 위해서는 이를 위한 추가적인 작업을 요구하게 됨으로 설계자들에게 새로운 부담을 주게 될 뿐만 아니라, 이와 같은 정식검증 기법은 검증 대상이 되는 설계의 크기가 커지게 되면 해당 기법이 상태 폭발(state explosion)의 문제로 인하여 컴퓨터 상에서 수행이 불가능해진다는 매우 큰 약점이 있다.

<15> 또 다른 문제점으로는 검증을 하기 위하여 설계 코드를 이용한 검증 실행이 일정 검증 사이클 동안(예로 시뮬레이션으로 검증을 수행하는 경우에 0 나노초 시뮬레이션 사이클에서부터 10,000,000 나노초 시뮬레이션 사이클까지) 이루어져야 하는데 이 과정에서 설계 코드의 수행이 설계자가 의도하는 대로 진행되는 가를 확인하는 과정이 반드시 필요하다. 이와 같은 확인 과정에서 필요한 것이 설계 코드에서 존재하는 시그널들이나 변수들에 대한 탐침(probing)인데, 이와 같은 탐침을 통하여 설계에 대한 가시도(visibility)가 확보되는 것이 설계 코드에 존재하는 오류들의 위치를 확인하고 제거하는 것에 반드시 필요하게 된다. 그러나 이와 같은 탐침에서의 문제점으로는 설계 코드에는 매우 많은 수의 시그널들이나 변수들이 존

재하게 되는데, 이들 중에서 상기의 일정 검증 사이클 동안에서 설계 코드를 이용한 검증 실행이 실제 수행되기 전에 설계 오류의 위치를 알아내고 이를 제거하기 위해서 탐침이 필요한 시그널들이나 변수들을 선정하는 것이 매우 어렵다는 것이다. 현재 사용되는 방법은 시뮬레이션 내지는 시뮬레이션가속 내지는 에뮬레이션 내지는 프로토타이핑 실행 전에 검증 대상에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정한 후에 상기 시뮬레이션 내지는 시뮬레이션가속 내지는 에뮬레이션 내지는 프로토타이핑을 실행하는 것이 한 방법이다. 그러나, 이와 같이 검증 대상에 존재하는 모든 시그널들과 변수들을 탐침대상으로 하면, 탐침대상을 정하지 않고 상기 시뮬레이션 내지는 시뮬레이션가속 내지는 에뮬레이션 내지는 프로토타이핑을 실행하는 것에 비하여 실행 속도가 최소 2배에서 최대 10배 이상 증가하게 되는 문제점이 있다.

<16> 또 다른 문제점으로는 수백만 게이트급 이상의 디지털 시스템을 효율적으로 검증하기 위해서는 시뮬레이션, 시뮬레이션가속, 에뮬레이션, 프로토타이핑, 정식검증들 중에서 두 개 이상의 검증기술을 같이 사용한 검증 수행 과정이 요구되는데, 실제 이들 기술들을 같이 사용하는 경우에도 단지 하나의 기술(예로 시뮬레이션)을 사용하여 최대한 설계 오류들을 찾아서 수정한 후에, 또 다른 기술(예로 정식검증)을 사용하여 남아있는 설계 오류들을 찾아서 수정하는 단순한 수준에서 두 개 이상의 검증기술을 같이 사용하는 수준에 머물고 있다.

<17> 뿐만 아니라, 설계코드 내지는 설계회로에 존재하는 특정한 1 이상의 설계오류들이 발견되어져서 제거되는 과정에서 설계코드 내지는 설계회로, 또는 필요시에

는 테스트벤치가 수정되어진다. 이와 같이 설계대상 객체들이 부분적으로 수정이 되어지면 수정되기 전에 실행되었던 검증 과정들에 대한 반복적인 수행을 통하여 후방향 호환성(back-ward compatibility)를 조사하는 리그레션 테스트(regression test) 과정이 필요하게 되는데 이와 같은 과정이 현재에는 매우 비효율적으로 진행되어짐으로 많은 시간을 소비한다.

<18> 뿐만 아니라, 설계대상 객체들이 디버깅을 위하여 수정이 되지 않은 경우에도 설계검증 과정에서는 여러 가지 검증 시나리오들을 여러 가지의 테스트 수트들로써(예로 1,000개의 각기 다른 테스트 수트들을 사용하여서) 수행하게 된다. 이 테스트 수트들중에서 많은 것들은 검증 실행 시작에서부터 일정 시간 동안에는 동일하게(예로 SOC의 경우 HW 초기화에 이은 SW 초기화 과정들은 여러 테스트 수트들에서 동일함) 진행되고, 이 후부터는 다르게 진행되는 경우가 많다. 이와 같은 경우에 동일하게 진행되는 과정은 한번만 실행하고 매 테스트 수트마다 반복하지 않는다면 매우 신속하게 여러 가지의 검증 시나리오들을 실행할 수 있는데 현재의 방법들은 그렇지 못하다.

【발명이 이루고자 하는 기술적 과제】

<19> 따라서, 본 발명의 목적은 초대규모급 디지털시스템 설계에 대한 검증을 위한 검증의 성능 향상과 효율성을 증대시키는 검증 장치 및 이를 이용한 신속한 검증 방법을 제공함에 있다. 구체적으로는 시뮬레이션, 시뮬레이션가속, 하드웨어에뮬레이션, 프로토타이핑 등에서 이전에 수행된 검증 결과들을 효과적으로 재활용함으로서 검증의 속도 향상과 더불어 효율성을 증대시키고, 결과적으로 연속적인 설

계 오류들에 대한 신속한 발견과 디버깅을 가능하게 한다.

【발명의 구성】

<20> 상기 목적들을 달성하기 위하여, 본 발명에 따른 검증 장치는 검증 소프트웨어와, 1 이상의 HDL/SDL 시뮬레이터가 인스톨된 1이상의 컴퓨터로 구성된다. 또는 본 발명에 따른 검증 장치는 검증 소프트웨어와, 1 이상의 HDL/SDL 시뮬레이터가 인스톨된 1이상의 컴퓨터와, 1 이상의 하드웨어에뮬레이터 내지는 1 이상의 시뮬레이션가속기 내지는 1 이상의 프로토타이핑시스템으로 구성된다. 또는, 본 발명에 따른 검증 장치는 검증 소프트웨어가 인스톨된 1이상의 컴퓨터와, 1 이상의 하드웨어 에뮬레이터 내지는 1 이상의 시뮬레이션가속기 내지는 1 이상의 프로토타이핑시스템으로 구성된다.

<21> 본 발명에서 제안되는 검증 장치와 검증 방법은 설계 코드 자체를 검증하는 함수적 검증(functional verification)에 사용될 수 있을 뿐만 아니라, 설계 코드를 합성한 게이트수준의 네트리스트를 이용한 게이트수준의 검증에서도 사용될 수 있고, 또는 배치(placement) 및 배선(routing)이 되고 추출된 타이밍정보를 게이트수준의 네트리스트에 첨부시켜(back-annotated) 수행하는 타이밍 검증에서도 사용될 수 있다. 그러나 앞으로의 설명은 설계 코드 자체를 검증하는 함수적 검증에 대하여 하기로 하며, 게이트수준 검증이나 타이밍 검증에 대해서도 같은 방법을 적용할 수 있음으로 구체적 설명은 생략하기로 한다.

<22> 검증 소프트웨어는 HDL/SDL(예로 Verilog, VHDL, System Verilog, System C 등)을 사용하여 작성된 설계 코드를 읽은 후에 여기에다 추가적인 코드를

부가한다. 부가된 코드는 기본적으로 설계 코드로써 시뮬레이션 내지는 시뮬레이션 가속 내지는 하드웨어에뮬레이션을 수행하는 과정에서 일정 간격으로 설계객체들에 대한 동적정보를 저장하는 역할을 수행하고, 또한 추후에 사용자의 요구에 따라서 저장된 설계객체들에 대한 동적정보를 이용하여 시뮬레이션 내지는 시뮬레이션가속 내지는 하드웨어에뮬레이션을 재개하게 한다. 설계객체들이란 크게 DUV 뿐만 아니라 테스트벤치(testbench, 이후에는 TB로 약칭함) 모두를 포함하고 있을뿐만 아니라, 이들 DUV와 TB는 일반적으로 계층적인 구조에 이들은 다양한 설계블럭(design block)들을 가지고 있는데, 이들 설계블럭들 각각도 설계객체라 할 수 있다. 설계객체들에 대한 동적정보란 검증수행 도중에 설계객체 상에서 시간적으로 변하게 되는 값들의 집합을 가르킨다. 예로는 설계코드의 경우에는 특정검증 시간대에 설계코드에 존재하는 모든 변수들의 값들이 될 수 있고, 설계회로의 경우에는 특정검증 시간대에 설계회로에 존재하는 모든 신호선들의 값들이 될 수 있다. 또한 특정 설계객체에 있어서 이것에 대한 동적정보 중에서도 설계객체에 존재하는 메모리소자들만에 대한 동적정보를 상태정보라 하고, 설계객체의 모든 입력과 양방향 입출력들만에 대한 동적정보를 재생용 입력정보라 한다.

<23> 이와 같은 동적정보를 검증 실행 도중에 설계객체로부터 얻기 위해서 사용되는 방법은 사용되는 검증 방법에 따라 다르다. 검증 방법이 시뮬레이션인 경우에는 시뮬레이션의 100% 가시도(visibility) 기능(예로 VCD dump)을 이용하여 검증 실행 도중에 원하는 동적정보를 얻을 수 있다. 검증 방법이 하드웨어플랫폼을 이용하는 시뮬레이션가속이나 에뮬레이션의 경우에는 설계객체에 입출력탐침 부가회로(참

조)를 부가하여 구현함으로서 검증 실행 도중에 원하는 동적정보를 입출력탐침을 통하여 얻을 수 있다. 혹은 상용 하드웨어플랫폼(예로 Cadence의 Palladium, Mentor의 Celaro/Vstattion, Verisity의 ExtremeII, Tharas의 Hammer 등)에서 제공하는 100% 가시도 기능을 이용하여서도 검증 실행 도중에 원하는 동적정보를 얻을 수 있다.

<24> 이상과 같은 방법으로 특정 검증실행 과정 V(t)에서 얻어진 동적정보는 이후에 실행되는 또 다른 검증실행 과정 V(t+j)에 부분적으로 내지는 전체적으로 재활용되어서 V(t+j)의 검증실행 과정을 빠르게 진행할 수 있게 한다. 구체적으로, 이와 같이 재활용될 수 있는 경우는 2가지 경우로 나눌 수 있다.

<25> 첫 번째는, 특정 설계객체에 설계오류가 존재하여 해당 설계객체가 디버깅 과정을 거쳐서 수정이 된 경우에 리그레션테스트 과정이 필요하게 되는데, 이와 같은 경우에 재활용을 하는 것이다. 이 경우에는 리그레션테스트의 진행에 있어서 특정 테스트 시나리오를 실행하는 경우에 DUV 전체에 대하여 검증을 진행하지 않고, 전체 검증시간의 1 이상의 일정 영역에서는(예로 검증시간 0에서부터 일정 검증시간 t_m까지는) 설계 수정이 된 1 이상의 실계객체들만을 대상으로한 리그레션테스트를 진행함으로서 빠른 속도로 리그레션테스트가 진행될 수 있도록 하고, 이 이외의 영역에서는 DUV 전체에 대하여 검증을 진행하도록 한다. 이와 같은 방법을 위해서 핵심적인 사항은 설계 수정이 된 1 이사의 설계객체들만을 대상으로한 리그레션테스트를 진행하여도 무방한 1 이상의 일정 영역을 전체 검증시간 영역에서 구분할

수 있어야 한다. 이는 설계 수정이전에 수행되는 검증실행 과정에서 설계객체들에 대하여 동적정보를 수집하여 이를 설계 수정이후에 수행되는 검증실행 과정에서 설계 수정이 이루어진 설계객체들에서 생성되는 동적정보들과 검증실행 과정에서 실시간으로 비교하여 처리함으로서 가능하게 된다.

<26> 구체적인 방법의 일 예를 설명하면 다음과 같다. 리그레션테스트 진행 과정의 각 테스트 시나리오 실행에서 검증시간 0에서부터 설계 수정이 이루어진 설계객체들의 출력과 출력모드시의 양방향 입출력의 값들을 설계 수정 이전의 통일 설계객체들의 출력과 출력모드시의 양방향 입출력의 값들과 검증실행 과정에서 실시간으로 비교하여 이들 두 값들이 동일하다면 상기 설계 수정이 이루어진 설계객체들만을 설계 수정 이전의 검증 실행에서 구하여 특정 형태(예로 TB화된 형태, 내지는 VCD 형태, 내지는 이진파일 형태)로 저장시킨 재생용 입력정보를 입력스티뮬러스로 사용해서 매우 빠르게 실행하고, 상기 두 값들이 달라지는 시점에서 설계 수정이 이루어진 설계객체들과 설계 수정이 이루어지지 않은 나머지 설계객체들을 합하여서 검증을 계속하게 되면 전체의 검증 시간을 줄이면서도 바르고 정확한 검증을 수행하는 것이 가능하다. 그런데, 이와 같이 상기 두 값들이 달라지는 시점에서 설계 수정이 이루어진 설계객체들과 설계 수정이 이루어지지 않은 나머지 설계객체들을 합하는 과정에서 설계 수정이 이루어지지 않은 나머지 설계객체들의 동적정보를 설계 수정이 이루어진 설계객체들의 동적정보와 같은 검증시간대로 맞추어 주는 것이 필수적이다. 이는 설계 수정 이전에 수행된 설계검증 과정에서 설계객체들에 대하여 일정 검증시간 간격으로 동적정보를 1회 이상저장(예로 총 20회 저장)시키게 하

고, 이와 같이 저장된 복수개의 동적정보를 이용하면 가능하다.

<27> 두 번째는, 설계객체에 대한 수정 없이 다른 테스트 시나리오를 이용한 테스트를 진행하는 경우에 이에 앞서서 진행된 테스트 시나리오를 이용한 테스트 결과를 재활용 하는 것이다. 이 경우가 첫 번째 경우와 다른 것은 설계변경이 이루어진 블록이 이 경우에는 TB인 반면에, 첫 번째 경우에는 DUV에 존재하는 설계블럭이라는 것 이외에는 없다. 따라서, 첫 번째 경우에 적용된 방법과 동일한 방법을 두 번째 경우에 적용하여 앞서서 진행된 테스트 시나리오를 이용한 테스트 결과를 재활용 함으로서 다른 테스트 시나리오를 이용한 테스트를 신속하게 진행할 수 있다.

<28> 상기 2가지 경우에 설계객체가 변경되기 전의 실행결과와 설계객체가 변경된 후의 실행결과가 검증시간 0에서부터 시작하여 처음으로 달라지는 검증시점 t_m을 최대한도로 신속하게 찾아내는 것도 매우 중요하다. 이를 위해서는 다음과 같은 방법을 사용하면 매우 효과적이다. 설계객체가 변경되기 전의 검증실행 과정에서 1이상의 시점에서 설계객체에 대한 동적정보를 저장(일 예로 상태정보는 일정간격으로 저장, 재생용 입력정보는 매 이벤트 발생시마다 혹은 매 사이클마다 저장)하여서 상태 정보가 저장된 2 이상의 시점들에서 독립적으로 병렬적으로 설계검증 실행을 재현할 수 있게 하고, 1 이상의 설계 객체가 수정된 이후에 설계 검증을 수행을 상기 상태 정보가 저장된 2 이상의 시점들에서 독립적으로 병렬적으로 진행하고 수정된 1 이상의 설계 객체의 출력과 출력모드시의 입출력 값들이 달라지는 1 이상의 검증시간들 중에서 검증 시간 0(검증 시간 0은 검증의 시작시점임)에 제일 가까운

검증시간이 t_m이 된다. 이와 같이 t_m을 찾는 과정을 병렬적으로 진행하게 되면 t_m을 찾는 시간을 t_m을 찾는 과정을 순차적으로 진행하는 것에 비하여 크게 단축시키는 것이 가능할 수 있다.

<29> 상기 목적 외에 본 발명의 다른 목적 및 이점들은 첨부한 도면을 참조한 실시 예에 대한 상세한 설명을 통하여 명백하게 드러나게 될 것이다.

<30> 도1 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 시뮬레이터를 갖는 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 일 예를 개략적으로 도시한 도면이다.

<31> 도2 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 시뮬레이터를 갖는 2 이상의 컴퓨터들과 이들 컴퓨터들을 연결하는 컴퓨터 네트워크로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면이다.

<32> 도3 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 시뮬레이터를 갖는 컴퓨터와 하드웨어검증플랫폼으로 구성된 본 발명에 관한 설계 검증 장치의 일 예를 개략적으로 도시한 도면이다.

<33> 도4 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 시뮬레이터를 갖는 2 이상의 컴퓨터들과 하드웨어검증플랫폼과 이들 컴퓨터들을 연결하는 컴퓨터 네트워크로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면이다.

<34> 도5 는 설계객체가 수정된 후에도 설계객체가 수정되기 전에 실행된 설계검

증 과정에서 얻어진 동적정보를 재활용하여서 설계객체가 수정된 후에 실행되는 설계검증 과정을 신속하게 진행하는 과정을 개략적으로 도시한 도면이다.

<35> 도6 은 설계객체가 수정된 전의 설계검증 실행 결과와 설계객체가 수정된 후의 설계검증 실행 결과가 최초로 달라지는 시점을 설계객체가 수정되기 전에 실행된 설계검증 과정에서 얻어진 1 이상의 동적정보를 병렬적으로 이용하여 신속하게 구하는 과정을 개략적으로 도시한 도면이다.

【발명의 효과】

<36> 상술한 바와 같이, 본 발명에 따른 검증 장치 및 이를 이용한 설계 검증 방법의 목적은 초대규모급 설계 검증을 수행하는 과정에서 설계객체가 변경되는 경우에 수행되는 설계 검증을 설계객체가 변경되기 전에 수행한 설계 검증 결과를 재활용하여 신속하게 진행될 수 있게 함으로서 전체의 검증 시간을 단축하며, 빠른 시간 내에 설계 코드에 존재하는 오류들의 위치를 정확히 찾아내어 수정하는 것이 가능하다.

<37> 이상 설명한 내용을 통해 당업자라면 본 발명의 기술사상을 일탈하지 아니하는 범위에서 다양한 변경 및 수정이 가능함을 알 수 있을 것이다. 따라서, 본 발명의 기술적 범위는 실시예에 기재된 내용으로 한정되는 것이 아니라 특허 청구의 범위에 의하여 정하여져야만 한다.

【특허청구범위】

【청구항 1】

검증 소프트웨어와 1 이상의 검증 플랫폼을 구비하는 설계검증 장치에 있어서,

상기 검증 소프트웨어는 설계 객체들로 구성되는 설계 코드나 설계 회로에 자동화된 방식을 통하여 부가 코드나 부가회로를 부가하여 상기 1 이상의 검증 플랫폼들에 설계 코드나 설계 회로를 부가 코드나 부가회로와 함께 구현하여 검증 실행 도중에 1 이상의 특정 시점들에서 동적 정보의 수집을 가능하게 하고, 이 수집된 동적 정보를 1 이상의 설계 객체들이 수정된 후에 실행되는 검증 실행에 재활용함으로서 1 이상의 설계 객체들이 수정된 후에 실행되는 검증 실행의 전체 시간을 단축시킬 수 있게 하는 설계 검증 장치.

【청구항 2】

검증 소프트웨어와 1 이상의 검증 플랫폼을 이용하여,

설계 객체들로 구성되는 설계 코드나 설계 회로에 자동화된 방식을 통하여 부가 코드나 부가회로를 부가하여 상기 1 이상의 검증 플랫폼들에 설계 코드나 설계 회로를 부가 코드나 부가회로와 함께 구현하여 검증 실행 도중에 1 이상의 특정 시점들에서 동적 정보의 수집을 가능하게 하고, 이 수집된 동적 정보를 1 이상의 설계 객체들이 수정된 후에 실행되는 검증 실행에 재활용함으로서 1 이상의 설계 객체들이 수정된 후에 실행되는 검증 실행의 전체 시간을 단축시킬 수 있게 하는

설계 검증 방법.

【청구항 3】

제 1 항 내지는 제 2 항에 있어서,

1 이상의 설계 객체의 변경에 의하여 설계 검증 실행 결과가 설계 객체 변경 전의 설계 검증 실행 결과와 검증 시간 측면에서 제일 처음으로 달라지는 시점을 찾는 방법을 변경된 1 이상의 설계 객체의 모든 출력과 출력모드의 양방향 입출력들 값들과 변경되기 전의 상기 1 이상의 설계 객체의 든 출력과 출력모드의 양방향 입출력들 값들과 순서적으로 비교하여 찾아내는 설계 검증 방법.

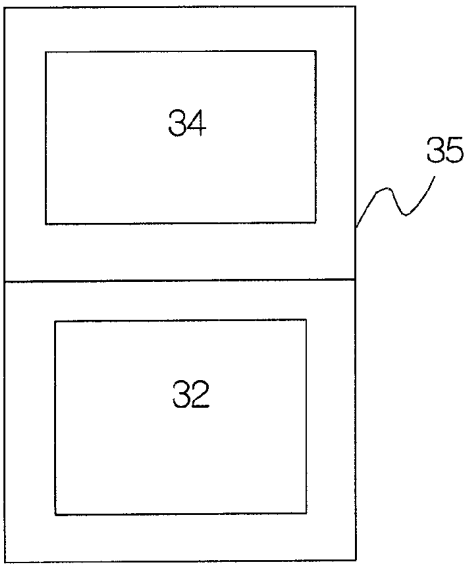
【청구항 4】

제 1 항 내지는 제 2 항에 있어서,

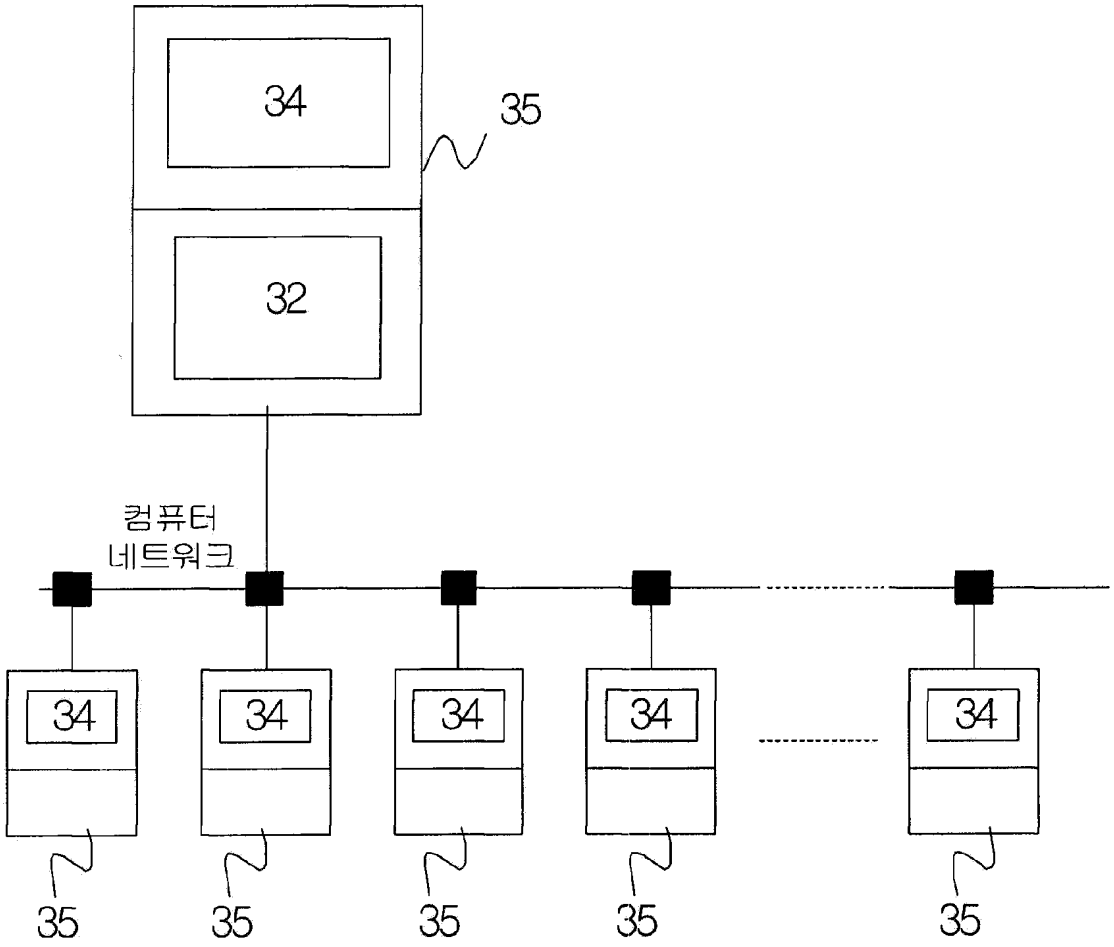
1 이상의 설계 객체의 변경에 의하여 설계 검증 실행 결과가 설계 객체 변경 전의 설계 검증 실행 결과와 검증 시간 측면에서 제일 처음으로 달라지는 시점을 찾는 방법을 변경되기 전의 설계 객체에 대한 1이상의 검증 시점에서의 동적정보를 병렬적으로 이용하여 찾아내는 설계 검증 방법.

【도면】

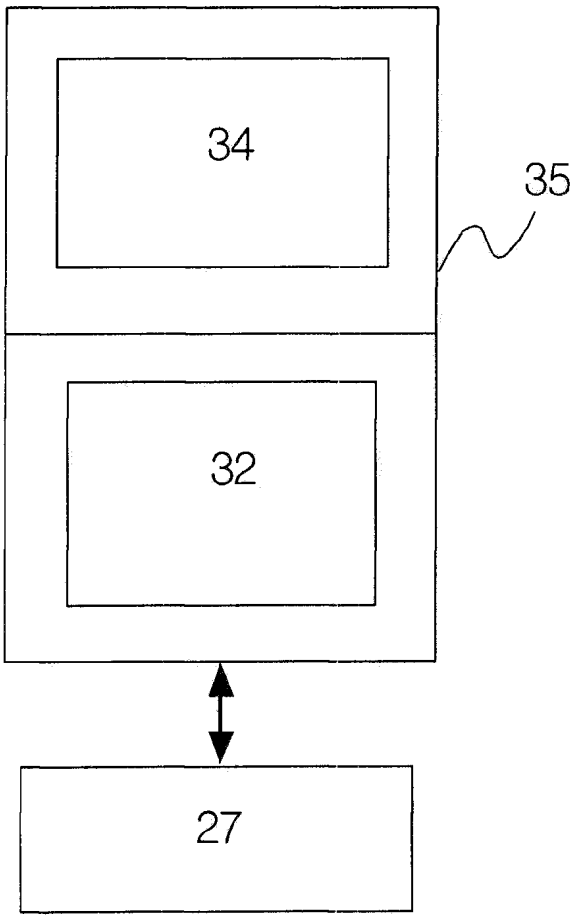
【도 1】



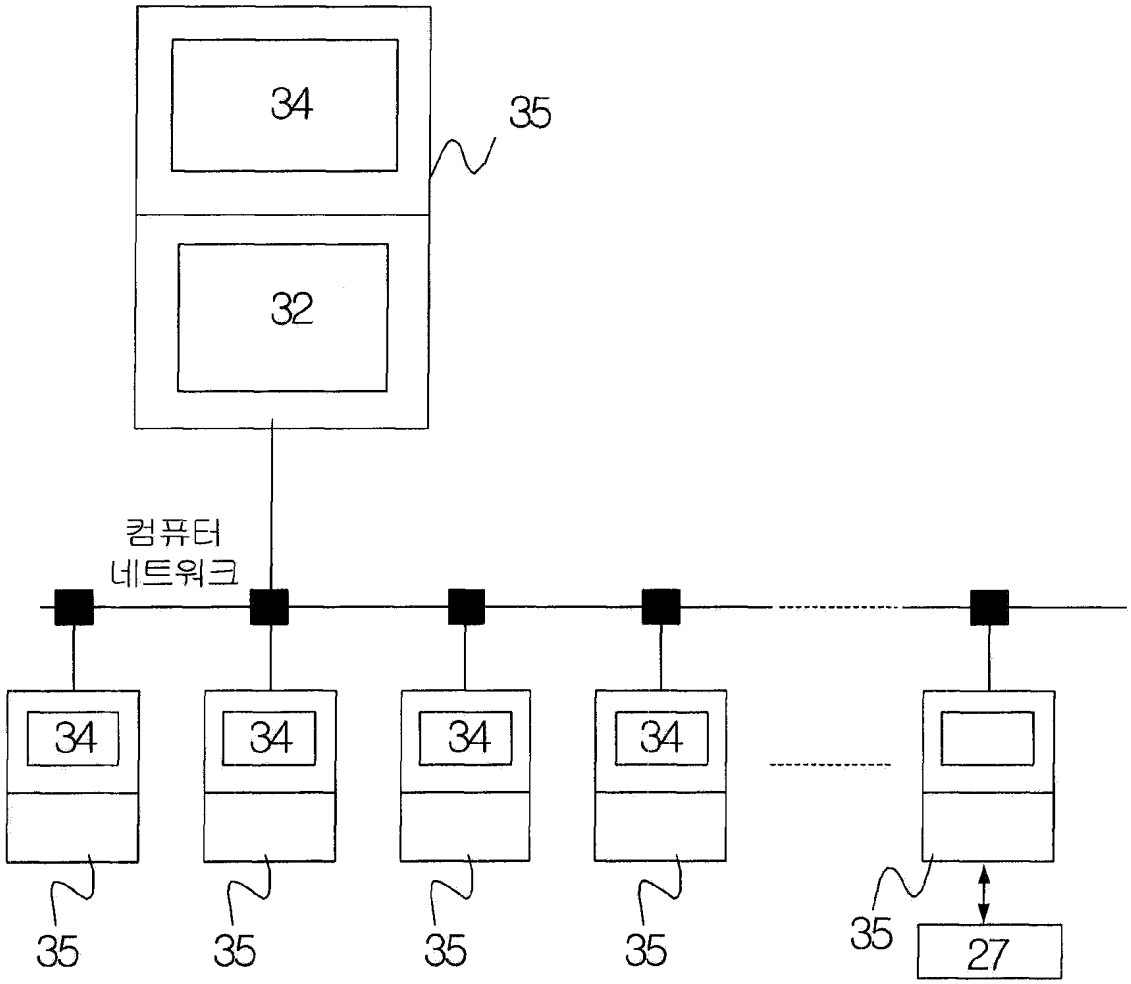
【도 2】



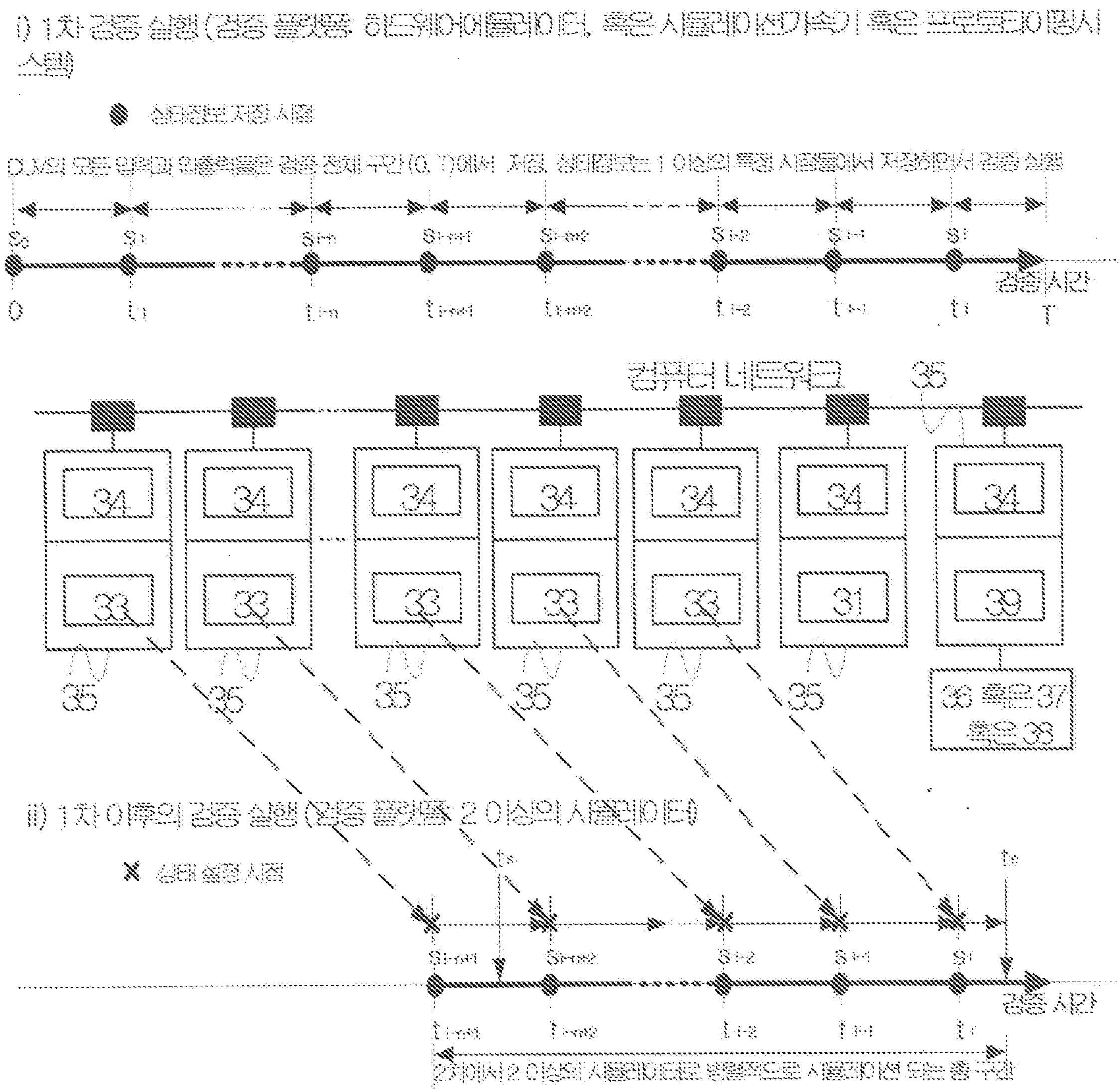
【도 3】



【도 4】



【도 5】



i) 1차 검증 실행(검증 플랫폼: 하드웨어에뮬레이터, 혹은 시뮬레이션가속기 혹은 프로토타이핑시스템)
상태정보 저장 시점
검증 시간
0
T
컴퓨터 네트워크
35
34
33
31
39
36 혹은 37 혹은 38
ii) 1차 이후의 검증 실행(검증 플랫폼: 2 이상의 시뮬레이터)
상태 설정 시점
검증 시간

【도 6】